Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/DE05/000266

International filing date: 16 February 2005 (16.02.2005)

Document type: Certified copy of priority document

Document details: Country/Office: DE
Number: 10 2004 007 835.1
Filing date: 17 February 2004 (17.02.2004)

Date of receipt at the International Bureau: 29 April 2005 (29.04.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

PCT/DE 2005/000266

BUNDESREPUBLIK DEUTSCHLAND

DE05/266

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen:	10 2004 007 835.1
Anmeldetag:	17. Februar 2004
Anmelder/Inhaber:	Sven W o o p , 66265 Heusweiler/DE
Bezeichnung:	Vorrichtung zur Darstellung von dynamischen komplexen Szenen
IPC:	G 06 T 1/20

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 18. April 2005
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Agurks

A 9161
06/00
EDV-L

Patentanmeldung

Vorrichtung zur Darstellung von dynamischen komplexen Szenen

Die Erfindung betrifft eine Vorrichtung, mit welcher dynamische, komplexe dreidimensionale Szenen mit hohen Bildwiederholraten unter Verwendung einer Echtzeit Ray-Tracing Hardwarearchitektur auf einem zweidimensionalen Display dargestellt werden können. Dynamische Szenen sind Szenen, in welchen sich neben der Kameraposition auch die Geometrie der darzustellenden Objekte von Frame zu Frame ändern kann. Die Erfindung zeichnet sich vor allem dadurch aus, dass sie eine Hierarchiestruktur von Objekten unterstützt, das heißt die Hauptszene kann aus mehreren Objekten bestehen, welche jeweils aus weiteren Objekten aufgebaut sind, wobei diese Schachtelung beliebig fortgeführt werden kann. Die auf den einzelnen Hierarchieebenen befindlichen Objekte können sowohl einzeln, als auch im Verbund bewegt werden. Hierdurch ist es möglich, komplexe Szenen mit hoher Dynamik zu erstellen und durch Verwendung des gleichen Objektes an mehreren Stellen der Szene die Repräsentation der Szene im Speicher klein zu halten.

Zur erfindungsgemäßen Realisierung dieser verschachtelten Objektebenen, wird die hardwaremäßige Umsetzung der bekannten Ray-Tracing-Pipeline um eine hardwaremäßig realisierte Transformationseinheit erweitert, welche die Strahlen in die Objekte hinein transformiert. Diese Einheit ist zur optimalen Ausnutzung der Hardwareressourcen nur einmal vorhanden und wird neben der Objektraumtransformation auch zur Berechnung des Schnittpunktes des Strahles mit einem Dreieck, der Erstellung von Primärstrahlen und zur Erstellung von Sekundärstrahlen verwendet.

Eine zweite Ausgestaltungsform der Erfindung gestattet dem Anwender die volle Programmierbarkeit des Systems, indem erfindungsgemäß eine neuartige Prozessorarchitektur, bestehend aus der Kombination eines Standard Prozessorkerns mit einem, oder mehreren speziellen Ray-Tracing-Coprozessoren verwendet wird.

Unter anderem zur Ausgabe der Bilddaten auf einem Display, kann die Erfindung durch die Verwendung gemeinsamer Frame-Buffer und Z-Buffer mit einer dem Stand der Technik entsprechenden Rasterisierungshardware kombiniert werden.

Stand der Technik

Der Stand der Technik bezüglich der Darstellung von dreidimensionalen Szenen ist derzeit in zwei Haupt-Sektoren einzuteilen, das Rasterisierungsverfahren und das Ray-Tracing-Verfahren (siehe Computer Graphics / Addison-Wesley ISBN 0201848406).

Das bekannte, vor allem in Computer-Graphikkarten zum Einsatz kommende Rasterisierungsverfahren beruht auf dem Prinzip, jede Geometrie der Szene auf einen Frame-Buffer und Z-Buffer zu projizieren. Hierzu werden die Farb- und Helligkeitswerte der Pixel im Frame-Buffer und die geometrischen Tiefenwerte im Z-Buffer gespeichert, jedoch nur dann, wenn der vorherige geometrische Wert im Z-Buffer größer (weiter vom Betrachter entfernt liegend) als der neue ist. Hierdurch wird sichergestellt, dass nähere Objekte fernere überschreiben und nach Ablauf des Algorithmus nur noch die wirklich sichtbaren Objekte im Frame-Buffer abgebildet sind.

Dieses Verfahren hat jedoch den entscheidenden Nachteil, dass aufwändige Szenen mit Millionen von Objekten mit heutiger Hardware nicht in Echtzeit dargestellt werden können, da es in der Regel erforderlich ist, alle Dreiecke (Objekte) der Szene zu projizieren. Desweiteren wird ein Frame-Buffer und Z-Buffer benötigt, auf welchem viele Milliarden Schreiboperationen in der Sekunde durchgeführt werden müssen, wobei zum Bildaufbau die meisten Pixel mehrfach pro Frame überschrieben werden. Das Überschreiben des zum Betrachter weiter entfernten Pixels durch Pixel näherer Objekte hat zur Folge, dass bereits berechnete Daten verworfen werden, wodurch eine optimale Systemleistung nicht realisiert werden kann.

Schatten sind wohl mit aufwändigen Techniken auf heutiger Rasterisierungshardware zu berechnen, jedoch nicht mit hoher Genauigkeit. Spiegelungen an gekrümmten Flächen, sowie die Darstellung von Lichtbrechungen sind mit dieser Technik nicht zu realisieren.

Leistungssteigernde Verbesserung schafft ein zweites Verfahren, das Ray-Tracing-Verfahren, welches durch seine photorealistischen Bilder, aber auch durch die Rechenkomplexität bekannt ist. Die Grundidee von Ray-Tracing steht in nahem Bezug zu physikalischen Lichtverteilungsmodellen (siehe Computer Graphics / Addison-Wesley ISBN 0201848406).

In einer realen Umgebung wird Licht von Lichtquellen emittiert und nach physikalischen Gesetzen in der Szene verteilt. Mit einer Kamera kann das Bild der Umgebung eingefangen werden. Ray-Tracing geht den umgekehrten Weg und verfolgt das Licht von der Kamera, welche die Betrachterposition darstellt, zurück zu ihrer Quelle. Hierzu wird für jedes Pixel des Bildes ein virtueller Strahl in die, das Pixel beleuchtende, Richtung geschossen. Dieses Schießen des Strahles nennt man Ray-Casting. Trifft der Strahl ein Objekt so berechnet sich die Farbe des Pixels unter anderem aus der Farbe des getroffenen Objektes, der Oberflächennormalen und den vom Auftreffpunkt sichtbaren Lichtquellen. Die sichtbaren Lichtquellen sind durch die Verfolgung der Sekundärstrahlen, welche von jeder Lichtquelle zu dem Auftreffpunkt geschossen werden, zu ermitteln. Treffen die Sekundärstrahlen ein Objekt zwischen Lichtquelle und Auftreffpunkt, so liegt der Punkt im Schatten bezüglich der Lichtquelle.

Dieses Verfahren ermöglicht neben der beschriebenen Schattenberechnung auch noch die Berechnung von Spiegelungen und der Brechungen des Lichtes, indem Reflexionsstrahlen bzw. gebrochene Sekundärstrahlen berechnet werden. Desweiteren können Szenen nahezu beliebiger Größe gehandhabt und dargestellt werden. Der Grund hierfür liegt in der Anwendung einer Beschleunigungsstruktur. Dies ist ein spezieller Algorithmus mit entsprechender Datenstruktur, welche es ermöglicht den virtuellen Strahl schnell durch die Szene zu „schießen" bzw. zu traversieren. Auf dem Weg werden einige Objekte selektiert, welche mögliche Trefferkandidaten sind, wodurch der Auftreffpunkt schnell gefunden wird. Theoretische Untersuchungen haben ergeben, dass die Komplexität des Ray-Tracing-Verfahrens im Mittel logarithmisch mit der Szenengröße wächst. Das heißt, eine Quadrierung der Zahl der Objekte der Szene bedeutet lediglich den doppelten Rechenaufwand.

Typische Beschleunigungsstrukturen sind zum Beispiel das regelmäßige Grid, der k-D Baum und der Octree (siehe Computer Graphics / Addison-Wesley ISBN 0201848406). Allen diesen Verfahren liegt die Idee zu Grunde den Raum in viele Teilräume aufzuteilen und für jeden solchen Teilraum die dort vorhandene Geometrie zu speichern. Der Traversierungs-Algorithmus verfolgt den Strahl dann von Teilraum zu Teilraum und schneidet diesen immer mit genau den Objekten, die sich in dem Teilraum befinden. Die drei Verfahren unterscheiden sich nur in der Anordnung der Teilräume. Bei dem regelmäßigen Grid ist der Raum in würfelförmige Teilräume gleicher Größe unterteilt Fig. 7. Die beiden anderen Verfahren beruhen auf einer rekursiven Unterteilung des Raumes. Beim k-D Baum Verfahren wird der Startraum rekursiv an einer beliebigen Stelle achsenparallel geteilt Fig. 8. Diese Aufteilung des Raumes wird in einer rekursiven Datenstruktur (hier ein binärer Baum) gespeichert. Das dritte Verfahren namens Octree, ist ebenfalls rekursiv, nur werden die betrachteten Teilräume immer in 8 gleichgroße rechteckige Teilräume unterteilt Fig. 9.

Im Gegensatz zum Rasterisierungsverfahren existiert momentan keine reine Hardwarelösung welche den Ray-Tracing-Algorithmus umsetzt, sondern nur softwarebasierende Systeme, welche enorm viel Rechenleistung und Rechenzeit erfordern. Zur Veranschaulichung des zeitlichen Umfanges der Berechnungen sei bemerkt, dass abhängig von der Komplexität des Bildes und der verwendeten Software mit der momentan dem Stand der Technik entsprechenden PC Hardware eine Rechenzeit von einigen Sekunden bis zu mehreren Stunden benötigt wird, um ein einzelnes Standbild nach diesem Verfahren zu erstellen. Die Berechnungen von Bewegtbildern erfordert entsprechend viel Rechenzeit und/oder die Verfügbarkeit von speziellen Großrechnern.

Der Lehrstuhl Computergraphik an der Universität des Saarlandes hat ein softwarebasiertes Echtzeit Ray-Tracing System entwickelt, welches auf einem Cluster von über 20 Rechnern zum Einsatz kommt.

Im US Patent 6,597,359 B1 ist eine Hardwarelösung für den Ray-Tracing-Algorithmus beschrieben, welcher sich jedoch auf statische Szenen beschränkt.

Das US Patent 5,933,146 beschreibt ebenfalls eine Hardwarelösung für den Ray-Tracing-Algorithmus, auch eingeschränkt auf statische Szenen.

Das Paper „ SaarCOR - A Hardware Architecture for Ray-Tracing " vom Lehrstuhl Computergraphik der Universität des Saarlandes beschreibt eine Hardwarearchitektur für Ray-Tracing jedoch wiederum limitiert auf statische Szenen.

Das Paper „A Simple and Practical Method for Interactive Ray-Tracing of Dynamic Scenes" vom Lehrstuhl Computergraphik der Universität des Saarlandes beschreibt einen Softwareansatz zur Unterstützung von dynamischen Szenen in einem Ray Tracer. Das beschriebene Software Verfahren verwendet jedoch nur eine Stufe von Objekten, kann also keine Schachtelung in mehreren Stufen durchführen.

Der beschriebene Stand der Technik bietet momentan weder Software- noch Hardwarelösungen, mit welchen komplexe dynamische Szenen in Echtzeit dargestellt werden können. Bei den bekannten Rasterisierungsalgorithmen liegt die Leistungsbegrenzung in der Zahl der darzustellenden Objekte.

Ray-Tracing Systeme können zwar viele Dreiecke darstellen, sind jedoch wegen der benötigten Vorberechnungen darin beschränkt, dass die Position nur eingeschränkt geändert werden kann. Szenen aus einigen Milliarden Dreiecken erfordern sehr viel Rechenleistung und Speicher und sind nur auf schnellen und komplexen Großrechnern oder Clusterlösungen zu handhaben.

Deshalb sind mit der verfügbaren Personal-Computerhardware softwarebasierte dynamische Echtzeit Ray-Tracing Systeme nicht realisierbar. Die beschriebene Clusterlösung dürfte aus Kostengründen auf Spezialanwendungen beschränkt bleiben.

Der Erfindung liegt die Aufgabe zugrunde, die Leistungsfähigkeit der bisher bekannten hardwarebasierten Rasterisierungsverfahren und der softwarebasierten Ray-Tracing-Verfahren derart zu verbessern, dass komplexe Objekte und dynamische Szenen in mehreren Schachtelungstiefen in einem dreidimensionalen Raum in Echtzeit dargestellt werden können.

Beschreibung der Erfindung

Gemäß Patentanspruch 1 und den Unteransprüchen wird diese Aufgabe wie folgt gelöst :

Vorrichtung zur Darstellung von dynamischen, komplexen Szenen, dadurch gekennzeichnet, dass diese aus einem oder mehreren programmgesteuerten Ray-Tracing-Prozessor/en besteht (Fig.5 / Fig.11), dessen/deren Funktionsumfang mittels hardwareimplementierter Traversierungs-Befehle und/oder Vektorarithmetikbefehle und/oder Befehle zur Erstellung von Ray-Tracing Beschleunigungsstrukturen realisiert wird

und/oder dass die Vorrichtung aus hardwarebasierten festverdrahteten oder verknüpften Logikfunktionen besteht (Fig.2 / Fig.3 / Fig.4), welche so organisiert sind, dass auf den Ray-Tracing-Verfahren basierende Funktionen bereitgestellt werden und zu deren Realisierung eine oder mehrere Traversierungs-Einheit/en und eine oder mehrere Listen-Einheit/en und eine oder mehrere Schnittpunktberechnungs-Einheit/en

und ein oder mehrere Transformations-Einheit/en und/oder eine oder mehrere Logikeinheit/en zum Lösen von linearen Gleichungssystemen gebildet werden

und die hardwaremäßige Gestaltung der Ray-Tracing-Prozessoren und/oder der Logikfunktionen in einem oder mehreren programmierbaren Logikbaustein/en oder in festverdrahteter integrierter oder diskreter Logik erfolgt.

Diese erfindungsgemäße, auf dem Ray-Tracing-Verfahren basierende Vorrichtung zur photorealistischen Darstellung dreidimensionaler bewegter Szenen, bei welcher die softwaremäßig definierten Beschleunigungsstrukturen, Algorithmen und Verfahren in entsprechende Hardwarestrukturen umgesetzt sind, ist vorrangig zum Echtzeiteinsatz vorgesehen.

Zur Realisierung beliebiger ungeordneter Dynamik in einer Szene muss für jedes Bild der Bildfolge die Beschleunigungsstruktur neu berechnet werden. Dies bedeutet bei großen Szenen einen enorm großen Rechenaufwand da die gesamte Geometrie der Szene „angefasst" werden muss. Hierbei verschwindet der Vorteil der logarithmischen Komplexität des Algorithmus in der Szenengröße.

Eine im Paper „A Simple and Practical Method for Interactive Ray Tracing" beschriebene Lösung zu diesem Problem ist die Unterteilung der Szene in Objekte und ausschließlich das Bewegen dieser Objekte als Ganzes zu erlauben. Hierbei werden zwei Beschleunigungsstrukturen benötigt.

Eine Top-Level Beschleunigungsstruktur über den Objekten der Szene und jeweils eine Bottom-Level Beschleunigungsstruktur für jedes der Objekte. Die Objekte werden hierbei in Form von Instanzen von Objekten in der Szene positioniert.

Der Unterschied eines Objektes zu der Instanz desgleichen liegt darin, dass eine Instanz eines Objektes aus einem Objekt und einer Transformation besteht. Die Transformation ist eine affine Funktion, welche das Objekt an eine beliebige Stelle der Szene verschiebt. Affine Transformationen erlauben desweiteren ein Skalieren, Rotieren und Scheren (engl. shearing) von Objekten. Im folgenden wird der Einfachkeit halber auch für Instanzen von Objekten der Begriff Objekt verwendet, falls keine Verwechslungsmöglichkeit besteht.

Ein Strahl wird zunächst durch die Top-Level Beschleunigungsstruktur traversiert (Strahl durch die Szene verfolgen) bis ein mögliches Treffer-Objekt (das vom Strahl getroffene Objekt) gefunden wird. Nun transformiert man den Strahl in das lokale Koordinatensystem des Objektes und traversiert in der Bottom-Level Beschleunigungsstruktur des Objektes weiter bis ein Treffpunkt mit einem Primitiven Objekt gefunden ist. Primitive Objekte sind Objekte, welche in sich keine weitere Struktur besitzen. Bei Ray Tracern sind das in der Regel Dreiecke und Kugeln.

Diese Methode funktioniert in der Praxis sehr gut, jedoch nur so lange die Zahl der Objekte nicht zu groß wird, da die Top-Level Beschleunigungsstruktur in jedem Bild neu aufgebaut werden muss. Das neue Aufbauen dieser Beschleunigungsstruktur ist erforderlich, wenn die Objekte in dieser bewegt wurden.

Die Erfindung stellt nun eine Hardwarelösung dar, welche obige Aufteilung der Szene in Objekte rekursiv unterstützt. Das heißt, sie schränkt sich nicht auf Objekte ein, welche aus Primitiven Objekten bestehen, sondern erlaubt ebenfalls, dass sich diese Objekte wieder aus Objekten zusammensetzen, welche wiederum aus Objekten bestehen können usw. Fig.1 zeigt wie aus mehreren Stufen von Objekten ein Baum erstellt werden kann. Zunächst wird als Objekt der Stufe 1 ein Blatt modelliert. Dieses Blatt wird nun mehrfach instantiiert und an einen Ast gesetzt, wodurch ein weiteres Objekt entsteht, jedoch jetzt ein Objekt der Stufe 2. Diese kleinen Äste können nun wieder mehrfach instantiiert werden zu einem größeren Ast oder Baum als Objekt der Stufe 3 usw. Es ist anzumerken, dass hier mehrere Ebenen von Objekten in Objekten vorkommen und dass die Repräsentation der Szene durch das mehrmalige Benutzen gleicher Geometrien klein ist.

Das in der Erfindung zum Einsatz kommende, durch entsprechenden Algorithmus definierte Verfahren für das Ray-Casting sieht wie folgt aus :

Der Strahl wird durch die Beschleunigungsstruktur der obersten Stufe traversiert bis ein mögliches Treffer-Objekt gefunden ist. Falls das Objekt ein Primitives Objekt ist, so wird der Schnittpunkt des Strahles mit dem Objekt berechnet. Ist das Objekt kein Primitives Objekt, so wird der Strahl in das lokale Koordinatensystem des Objektes transformiert und setzt dort die Traversierung rekursiv fort.

Ein wesentlicher Teil des Algorithmus ist die Transformation des Strahles in das lokale Koordinatensystem des Objektes, wodurch im Prinzip die Positionierung des Objektes durch die affine Transformation rückgängig gemacht wird. Das heißt, der transformierte Strahl sieht das Objekt nun nicht mehr transformiert. Dieser Transformationsschritt erfordert eine recht aufwändige affine Transformation des Strahlstartpunktes und der Strahlrichtung, wobei jedoch die dazu erforderliche aufwändige Hardwareeinheit zusätzlich noch für weitere Aufgaben einsetzbar ist. Es stellt sich heraus, dass die Transformationseinheit ebenfalls zur Berechnung des Schnittpunktes mit vielen Arten von Primitiven Objekten, zur Berechnung von Primärstrahlen und zur Berechnung vieler Arten von Sekundärstrahlen verwendet werden kann.

Zur Berechnung der Primärstrahlen wird eine ähnliche Kameratransformationsmatrix angewandt, wie bei den bekannten Rasterisierungsverfahren. Zunächst werden Pre-Primärstrahlen der Gestalt R=((0,0,0),(x,y,1)) , also Strahlen mit dem Startpunkt (0,0,0) und der Richtung (x,y,1) definiert, wobei x und y die Koordinaten des Pixels zu dem ein Primärstrahl berechnet werden soll darstellt. Zu jeder Kameraposition und Ausrichtung gibt es eine affine Transformation, welche den Strahl R derart transformiert, dass er genau der Einfallsrichtung des Pixel (x,y) der Kamera entspricht.

Um den Schnittpunkt mit einem Primitiven Objekt zu berechnen, wird der Strahl in einen Raum transformiert, indem das Primitive Objekt normiert ist. Im Falle eines Dreiecks als Primitives Objekt, wird der Strahl beispielsweise derart in einen Raum transformiert, dass das Dreieck die Gestalt Δ_{norm}=((1,0,0),(0,0,0),(0,1,0)), wie in Fig. 10 zu sehen, hat. Diese Transformation kann durch eine affine Transformation geschehen. Die anschließende Schnittpunktberechnung mit dem Normdreieck ist im Gegensatz zum allgemeinen Fall sehr einfach in Hardware zu lösen. Wird die Transformation derart gewählt, dass die Dreiecksnormale auf den Vektor (0,0,1) im Dreiecksraum transformiert werden, so lässt sich das Skalarprodukt aus Strahl und Dreiecksnormale sehr einfach im Dreiecksraum berechnen, da das Skalarprodukt aus Strahlrichtung (xt,yt,zt) und der Dreiecksnormalen (0,0,1) gerade 0*xt+0*yt+1*zt = zt ist.

Die Transformation kann desweiteren so gewählt werden, dass nur 9 Fließkommazahlen für deren Repräsentation benötigt werden, indem die Dreiecksnormale auf eine geeignete Normale im Norm-Dreiecksraum abgebildet wird. Dies verhindert jedoch die Möglichkeit das Skalarprodukt im Norm-Dreiecksraum zu berechnen.

Es ist klar ersichtlich, dass diese Normobjekttransformation auch für andere Arten von Objekten verwendet werden kann, wie beispielsweise Kugeln, Ebenen, Quader, Zylinder und viele weitere geometrische Gebilde, es ist jeweils nur eine andere Schnittpunktberechnungseinheit zu erstellen.

Ein großer Vorteil hiervon ist die Tatsache, dass jede Art von Primitivem Objekt die gleiche Repräsentation im Speicher besitzt, nämlich eine affine Transformation welche in den Objektnormraum transformiert. Dies erleichtert die Konzeption des Speicherinterface einer Hardwarelösung. Die Transformation in den Objektnormraum nennt man die Normraumtransformation.

Lichtstrahlen sowie Spiegelungen lassen sich durch die Berechnung geeigneter Transformationen und geeigneter Strahlen effizient durch die Transformationseinheit berechnen.

Desweiteren ist es möglich, mit der Transformationseinheit Normalen (Vektoren die senkrecht auf einer Fläche stehen) zu transformieren. Diese Normalentransformation ist deswegen nötig, da einige Shadingmodelle die Normale der Geometrie am Auftreffpunkt benötigen. Diese Normale muss jedoch im Weltkoordinatensystem vorliegen, was bei obigem Algorithmus nicht zwangsläufig der Fall ist. Vielmehr liegt die Normale erstmals nur im lokalen Koordinatensystem des getroffenen Objektes vor. Sie muss von dort wieder zurück in das Weltkoordinatensystem transformiert werden.

Die Transformationseinheit hat jedoch auch einen Nachteil. Da die affinen Transformationen, welche als Matrizen gespeichert werden können, sowohl für Dreiecke also auch für die Objekte der Szene vorberechnet werden müssen, ist es nicht ohne weiteres möglich die Position der Dreieckseckpunkte effizient von Frame zu Frame zu ändern. Dies ist in Vertexshadern auf heutigen Grafikkarten jedoch möglich. Vertexhader sind programmierbare Spezialeinheiten, welche darauf optimiert sind, Bewegungen von Punkten im Raum zu berechnen.

Um dies zu ermöglichen, muss man sich von der Vorberechnung der Daten lösen. Demzufolge ist es dann erforderlich zur Schnittpunktsberechnung mit einem Dreieck ein lineares Gleichungssystem mit drei Unbekannten zu lösen. Dieses explizite Lösen erfordert zwar mehr Fließkomma-Operationen, ist jedoch in Verbindung mit Vertexshadern erforderlich. Demzufolge kann es sinnvoll sein, obige Transformationseinheit durch eine Einheit, welche ein lineares Gleichungssystem löst, zu ersetzen. Diese Einheit kann unter anderem dazu verwendet werden, um mit Dreiecken zu schneiden oder Strahlen in das lokale Koordinatensystem eines Objektes zu transformieren.

Ein Problem von detailreichen Szenen sind unerwünschte Alaising Effekte, die vor allem dann entstehen, wenn die Objektdichte in einer Richtung sehr hoch ist. Dann kann es passieren, dass ein Strahl beispielsweise ein schwarzes Dreieck trifft und bei einer minimalen Bewegung der Kamera plötzlich ein weißes Dreieck getroffen wird. Solche Effekte führen zu einem zeitlichen und örtlichen Rauschen im Bild. Der Grund liegt darin, dass Ray-Tracing in der Regel unendlich schmale Strahlen verwendet und nicht berücksichtigt, dass sich das Licht, welches einen Pixel beeinflusst, pyramidenförmig ausbreitet und sich der Strahl mit der Entfernung aufweitet. So müssten eigentlich alle Objekte, welche sich in dieser Strahlenpyramide befinden, zur Berechnung der Pixelfarbe herangezogen werden, was in einem Echtzeitsystem nicht möglich ist. Abhilfe schafft hier eine neue vereinfachte Form des Cone-Tracings. Anstatt einen beliebig schmalen Strahl zu betrachten, wird zusätzlich der Öffnungswinkel des Strahles bewertet. So kann je nach Entfernung zur Kamera die entsprechende Strahlbreite berechnet werden. Trifft man bei der Traversierung auf einen Teilraum der zu einem Großteil von dem Strahl überdeckt wird, so macht es unter Umständen keinen Sinn mehr weiter zu traversieren. Optimal ist an dieser Stelle eine vereinfachte Geometrie des Volumeninneren zur Berechnung zu verwenden und zu ignorieren, dass in dem Volumina vielleicht eine Million Dreiecke sind. Diese Dreiecke bilden unter Umständen nur die Wand eines Gebirges, welche wegen der Größe des Strahles auch durch eine farbige Ebene approximiert werden kann.

Falls die Dreiecke jedoch ein löchriges Gebilde wie zum Beispiel den Eifelturm modellieren, ist als Approximation eher die Farbe der konstruktiven Gitterelemente und ein Transparenzwert zu wählen.

Die Erfindung unterstützt solche vereinfachten Geometriepräsentationen in der Beschleunigungsstruktur. Fig. 6 zeigt das Konzept am Beispiel eines Octrees. Zu dem fett umrandeten Volumina, zu welchem ein Knoten in der Beschleunigungsstruktur gehört, ist die vereinfachte Geometrie abgebildet. Der Strahl überlappt fast mit dem gesamten Volumen des Knotens, so dass die Vereinfachte Geometrie zur Schnittpunktsberechnung verwendet wird.

Eine alternative Methode besteht darin, die Objekte der Szene mit unterschiedlichen Detaillevel abzuspeichern. Das heißt, die Objekte mit unterschiedlicher Auflösung oder Anzahl Dreiecke zu modellieren und abhängig von der Entfernung der Objekte zur Kamera, detailreiche oder vereinfachte Objekte zu verwenden.

Ein Nachteil der oben beschriebenen festverdrahteten, hardwarebasierten Ray-Tracing-Pipeline ist die Tatsache, dass es aufwändig ist, sie programmierbar zu gestalten. Im Vergleich zu einem Software Ray-Tracing Ansatz, wirkt die Hardwarebasierte Pipeline sehr starr und speziell. Abhilfe schafft die Entwicklung einer speziell auf den Ray-Tracing-Algorithmus angepaßten CPU. Diese spezielle Ray-Tracing Processing Unit abgekürzt RTPU besteht aus einer Standard CPU z.B. RISC Prozessor, dessen Befehlssatz um spezielle Befehle erweitert wird. Insbesondere ist ein Traversierungs-Befehl wichtig, welcher den Strahl durch eine Beschleunigungsstruktur traversiert. Einige Stellen des Algorithmus erfordern desweitern aufwändige arithmetische Operationen, welche vorwiegend im dreidimensionalen Raum geschehen. Es ist demnach sinnvoll die CPU mit einer Vektorarithmetikeinheit auszustatten, ähnlich den heute geläufigen SSE2 Befehlssätzen. Eine weitere Optimierung der CPU kann dadurch erreicht werden, indem die Parallelisierbarkeit des Algorithmus ausgenutzt wird. Es ist demnach möglich, sehr effektiv mehrere Threads (Programmabläufe) auf einer CPU laufen zu lassen, was die Auslastung und Effektivität der CPU dramatisch erhöht, vor allem im Bezug auf die Speicherwartezeiten.

Macht ein Thread eine Speicheranfrage, so kann ein anderer während der Anfrage ausgeführt werden. Ein beispielhafter Aufbau solch einer CPU ist in Fig. 5 zu sehen. Da bei dynamischen Szenen für jedes Frame die Neuberechnungen der umfangreichen Beschleunigungsstrukturen durchgeführt wird, ist es erforderlich, den Befehlssatz der CPU um spezielle Befehle zur Erstellung der Beschleunigungsstrukturen zu erweitern.

Die Einheiten der Ray-Tracing Architektur benötigen eine sehr hohe Speicherbandbreite, das heißt es müssen sehr viele Daten pro Zeiteinheit übertragen werden. Normalerweise ist dies nur zu realisieren, indem sehr viele Speicherchips parallel geschaltet werden. Die erforderliche Speicherbandbreite kann jedoch auch durch eine geeignete Verschaltung von mehreren Cachestufen (n-Level Caches) sichergestellt werden. Essentiell ist hier eine Eigenschaft des Ray-Tracing-Algorithmus die man Kohärenz nennt. Kohärenz bezeichnet die Tatsache, dass Strahlen die ähnliche Bereiche des 3D Raumes durchlaufen auch auf nahezu die gleichen Daten in der Beschleunigungsstruktur, dementsprechend also auch auf die gleichen Objekte zugreifen. Wird diese Eigenschaft ausgenutzt, können hohe Cache Hitraten erzielt werden. Das heißt, die benötigten Daten werden mit großer Wahrscheinlichkeit im Cache wiedergefunden und brauchen nicht zeitaufwändig aus dem Hauptspeicher geladen zu werden. Die Caches an sich sind wie in Fig. 4 gezeigt, in einem binären Baum angeordnet, um mehrere Ray-Tracing Einheiten zu versorgen.

Die erfindungsgemäße Vorrichtung kann in Verbindung mit einem 3D Display natürlich auch zur photorealistischen dreidimensionalen Echtzeitdarstellung komplexer bewegter Szenen verwandt werden. Abhängig von der Technologie des eingesetzten Displays sind hierbei drei Ausführungsformen der Bildausgabe zu unterscheiden. Erstens eine Ausführungsform, bei welcher abwechselnd zwei den Stereoeindruck beinhaltende Bilder horizontal versetzt im Zeitmultiplexverfahren auf einem Display dargestellt werden.

Zweitens eine Ausführungsform bei welcher zwei den Stereoeindruck repräsentierende, horizontal versetzte Bilder, welche in abwechselnden senkrechten, die Bildinformation der beiden Bilder beinhaltende Streifen auf einem Display dargestellt werden. Drittens eine Ausführungsform, bei welcher die beiden horizontal versetzten Bilder auf zwei getrennten Displays gleichzeitig oder im Zeitmultiplexverfahren dargestellt werden.

Die beiden horizontal versetzten Bilder, welche jeweils dem rechten oder linken Auge zuzuordnen sind, werden durch entsprechend räumliche Displayanordnungen oder durch den Einsatz von Bildtrennvorrichtungen (z.B. Shutterbrillen, streifenförmige Fresnel-Prismen/Linsen, Polarisationsfilter) jeweils nur einem Auge sichtbar. Die einzusetzenden 3D Displays und deren Erfordernisse der Videosignalansteuerung entsprechen dem Stand der Technik und werden nicht näher beschrieben. Weitere Ausführungen zum Stand der Technik von 3D Displays sind folgenden, beispielhaft genannten Schriften zu entnehmen :

Computer Graphics / Addison-Wesley ISBN 0201848406, DE 4331715, DE 4417664, DE 19753040, DE 19827590, DE 19737449

Der Einsatz computeranimierter photorealistischer Echtzeitdarstellung dreidimensionaler bewegter Szenen und Bilder erstreckt sich über die Darstellung dreidimensionaler CAD Daten, die Darstellung medizinischer und technisch-analytischer Daten, über die Filmanimation sowie dem Einsatz in Flug- und Fahrsimulatoren, bis zu den sogenannten Home Anwendungen in Computerspielen mit aufwändiger Echtzeitgraphik.

Die funktionelle Realisierung und die Hardwareimplementierung der Ray-Tracing-Algorithmen und Verfahren erfolgt in komplexen und schnellen Logiktechnologien, wobei deren Umsetzung sowohl als festverdrahtete Digitallogik in Form von diskreter Digitallogik, oder Kunden- oder Anwendungsspezifisch gefertigten IC beispielsweise ASIC, oder von komplexen programmierbaren Logikbausteinen / Logikschaltkreisen, beispielsweise CPLD oder FPGA Technologien mit oder ohne CPU Kern erfolgt.

Beschreibung einer erfindungsgemäßen Ausbildungsform in hardwaremäßig realisierter Logik

Die nachfolgend beschriebene beispielhafte Ausbildungsform der Erfindung beschreibt die Ray-Tracing Einheit einer Computer- Graphikkarte, bei welcher die Hardwareimplementierung der Ray-Tracing-Algorithmen und Verfahren beispielsweise in einem freiprogrammierbaren Logikbaustein FPGA, in ASIC Technologie, oder in einem festverdrahteten Spezial Chip erfolgen kann.

Soweit Verfahren und Funktionsabläufe beschrieben werden, so sind diese rein hardwaremäßig zu realisieren. Was bedeutet, dass entsprechende Logikeinheiten und hardwaremäßig realisierte Arithmetikeinheiten zu gestalten sind.

Die Standardfunktion der Ansteuerelektronik zur Ansteuerung des Datendisplays (Kathodenstrahlröhre, TFT-, LCD- oder Plasmamonitor) und deren Timing entsprechen dem Stand der Technik, werden als bekannt vorausgesetzt und sind nicht Gegenstand der Beschreibung. Eine Schnittstelle zwischen dem Bildspeicher dieser Standardfunktion und der erfindungsgemäßen Umsetzung der Ray-Tracing-Algorithmen und Verfahren wird beschrieben.

Die Beschreibung gliedert sich in zwei Teile. Zunächst wird die Ray-Casting-Pipeline abgekürzt RCP beschrieben. Dabei handelt es sich um den Kern des Designs, welcher Strahlen durch die Szene traversiert und den Auftreffpunkt zurückliefert.

Im zweiten Teil wird für die Ray-Casting-Pipeline eine optimierte Ray-Tracing Architektur beschrieben, in welcher mehrere dieser Ray-Casting-Pipelines zusammenarbeiten.

Fig. 2 zeigt die Ray-Casting-Pipeline Einheit (RCP), welche aus mehreren Untereinheiten besteht. Die Traversierungs-Einheit traversiert den Strahl durch eine geeignete Beschleunigungsstruktur, vorzugsweise einen k-D Baum. Der Strahl wird so lange traversiert, bis er in einen Bereich der Szene kommt, in welchem sich mögliche Treffer-Objekte befinden. Die Objekte dieses Bereiches, sind in einer Liste gespeichert, welche von der Listen-Einheit bearbeitet wird.

Diese Objekte müssen nun auf einen möglichen Schnittpunkt hin untersucht werden und falls es keinen gültigen Schnittpunkt gibt, mit der Traversierung fortgesetzt werden. Die Listen-Einheit sendet die möglichen Treffer-Objekte, eins nach dem anderen, zur Matrix-Lade-Einheit, welche die zu dem Objekt gehörige affine Transformation lädt. Diese affine Transformation kann durch eine 4x3 Matrix dargestellt werden. Es kann sich dabei um Objektraumtransformationsmatrizen oder um Matrizen welche in den Normraum eines Primitiven Objektes transformieren, handeln. Nachdem die Matrix-Lade-Einheit die Matrix in der Transformationseinheit gespeichert hat, werden die Strahlen von der Strahl-Einheit durch die Transformationseinheit geschickt.

Nach der Transformation sind nun zwei Szenarien möglich. Zum einen kann es sich um ein Objekt handeln, welches noch weitere Objekte beinhaltet. Ist dies der Fall, so wandern die Strahlen wieder zurück in die Traversierungs-Einheit und der transformierte Strahl wird in dem Objekt weiter traversiert. Handelt es sich jedoch um ein Primitives Objekt, so geht der Strahl direkt weiter in die Schnittpunktberechnungs-Einheit, welche den Stahl mit dem Normobjekt schneidet. Die Schnittpunktberechnungs-Einheit kann wie zuvor beschrieben mehrere Normobjekte (Dreiecke, Kugeln usw.) unterstützen.

Die berechneten Schnittpunkt-Daten werden in der Schnittpunkt-Einheit gesammelt. Die Schnittpunkt-Einheit liefert einen Rückkanal zur Traversierungs-Einheit, so dass diese erkennen kann, ob schon gültige Schnittpunkt-Daten vorhanden sind.

Die Traversierungs-Einheit, Listen-Einheit und Matrix-Lade-Einheit sind die einzigen Einheiten der Ray-Casting-Pipeline, welche auf externen Speicher zugreifen. Die Traversierungs-Einheit greift auf die Beschleunigungsstruktur zu, die Listen-Einheit auf Listen von Objektadressen und die Matrix-Lade-Einheit auf affine Transformationen in Form von Matrizen. Alle drei Einheiten sind über einen eigenen Cache mit dem Hauptspeicher verbunden, um die nötige Speicherbandbreite zu gewährleisten.

Eine vereinfachte Version der Ray-Casting-Pipeline ist in Fig. 12 dargestellt, wobei nur die wichtigsten Einheiten abgebildet sind : die Traversierungs-Einheit welche die Strahlen durch die Beschleunigungsstruktur traversiert, die Listen-Einheit, welche die Listen bearbeitet, die Transformations-Einheit, welche die geladene Transformation auf die Strahlen anwendet, und die Schnittpunktberechnungs-Einheit, welche den transformierten Strahl mit dem Norm-Objekt schneidet.

Die Ray-Casting-Pipeline ist wie in Fig. 3 gezeigt in eine geeignete Ray-Tracing Architektur eingebettet. Die Abbildung zeigt 4 Ray-Casting-Pipeline Einheiten mit ihren jeweils 3 Caches. Es sind hier jeweils 2 Einheiten mit einer Shading Einheit verbunden. Diese Shading Einheiten verwenden die Ray-Casting-Pipeline Einheiten um die Farben der Pixel des Bildes zu berechnen. Hierzu schießt die Shading Einheit Primärstrahlen, verarbeitet die Auftreffinformationen, welche die Ray-Casting-Pipeline zurückliefert und schießt Sekundärstrahlen beispielsweise zu Lichtquellen.

Die Shading Einheiten besitzen einen Kanal zur Transformationseinheit der Ray-Casting-Pipeline. Dieser wird verwendet um Kameramatrizen und Matrizen für Sekundärstrahlen zu laden und dadurch den Rechenaufwand für die Shading Einheit zu minimieren. Anzumerken ist, dass die Shading Einheiten jeweils einen getrennten Textur Cache und Shading Cache besitzen. Der Shading Cache enthält Shadinginformationen zu der Geometrie der Szene wie zum Beispiel Farben und Materialdaten. Der Texturcache ist mit dem Texturspeicher verbunden und ermöglicht den Shading Einheiten Zugriff auf Texturen. Jede Shading Einheit besitzt ihren eigenen lokalen Frame-Buffer, auf welchem sie die Farben und Helligkeitswerte der gerade bearbeiteten / berechneten Pixel ablegt. Desweitern ist ein Z-Buffer vorhanden, welcher für die nachfolgend beschriebene Anbindung an die Standardrasterisierungshardware benötigt wird.

Ist die Farbe eines Pixels von der Shading Einheit vollständig berechnet, so wird diese Farbe über die Tone-Mapping-Einheit in den globalen Frame-Buffer geschrieben. Die Tone-Mapping-Einheit wendet eine einfache Funktion auf die Farbe an, um diese in dem 24 Bit RGB Raum abzubilden. Auch die geometrischen Tiefenwerte (Z-Werte), welche im lokalen Z-Buffer gespeichert werden, werden nun in den globalen Z-Buffer übertragen.

Die Farbe bzw. der neue Z-Wert werden jedoch nur dann in den Frame-Buffer bzw. Z-Buffer geschrieben, wenn der zuvor im Z-Buffer vorhandene Z-Wert größer ist. Hierdurch ist sichergestellt, dass nur dann Pixel geschrieben werden, wenn sie geometrisch vor bereits von der Rasterisierungshardware oder anderen Ray-Tracing Passes berechneten Werten liegen. So ist es möglich die Ray-Tracing Hardware mit einer Standard Rasterisierungshardware zu kombinieren, welche auf dem gleichen Frame-Buffer bzw. Z-Buffer arbeitet, welcher hierbei auch die Schnittstelle zu dieser Standard Rasterisierungshardware darstellt.

Zur weiteren Erhöhung der Systemleistung können optional noch weitere Shading-Einheiten mit den zugehörigen Ray-Casting-Pipelines und Caches parallelgeschaltet werden. Der leistungssteigernde Effekt liegt hierbei in der Verbreiterung der Daten- und Verarbeitungsstruktur.

Die Signalaufbereitung und die Timinggenerierung für das Display bzw. den Monitor erfolgt in bekannter Art durch diese Rasterisierungshardware. Hierbei werden die Grundfunktionen der Standard Rasterisierungshardware mit den hardwareimplementierten Ray-Tracing-Algorithmen und Funktionen zu einer sehr leistungsfähigen Echtzeit Hardwarearchitektur verbunden.

Beschreibung der zweiten Ausbildungsform der Erfindung

als programmierbarer Ray-Tracing-Prozessor

Die zweite beispielhafte Ausbildungsform der Erfindung basiert auf der Gestaltung und Anwendung von frei programmierbaren Ray-Tracing CPU's abgekürzt RTPU, welche programmgesteuert die erfindungsgemäß beschriebenen, speziellen Ray-Tracing Funktionen und Algorithmen ausführen. Hierbei werden durch entsprechende Logik- und Funktionsparallelität jeweils nur wenige, vorzugsweise ein oder zwei CPU Taktzyklen zur Abarbeitung der Einzelfunktion benötigt.

Soweit interne Algorithmen, Verfahren und Funktionsabläufe der Ray-Tracing CPU abgekürzt RTPU beschrieben werden, sind diese mittels einer Hardware-beschreibungssprache, zum Beispiel HDL, VHDL, oder JHDL zu erstellen und auf die entsprechende Hardware zu übertragen. Die Hardwareimplementierung der RTPU mit den implementierten Algorithmen und Verfahren kann beispielsweise in einem freiprogrammierbaren Logikbaustein FPGA, in ASIC Technologie, in der Kombination von digitalen Signalprozessoren mit FPGA / ASIC oder in einem festverdrahteten Spezial Chip erfolgen.

Dies bedeutet, dass entsprechende Logikeinheiten und hardwaremäßig realisierte Arithmetikeinheiten zu gestalten sind, deren Einzelfunktionen programmgesteuert abgerufen werden können. Abhängig von der Komplexität der eingesetzten Hardwaretechnologie kann pro Chip eine oder mehrere RTPUs, mit oder ohne weiteren Logikfunktionen realisiert werden.

Hierbei werden wie in Fig. 11 zu sehen ist, mehrere der in Fig. 5 dargestellten RTPU Einheiten parallel geschaltet. Das Speicherinterface wird von einer Cache-Hierarchie gebildet, welche die nötige Speicherbandbreite bereitstellt. Dieses Vorgehen ist wiederum nur wegen der starken Kohärenz benachbarter Strahlen effizient möglich. Bei dem RTPU Konzept bearbeitet jede RTPU entweder genau ein Pixel des Bildes oder ein Paket von mehreren Pixeln.

Die Einzelpixelbearbeitung ist wegen ihrer Einfachheit vorzuziehen, benötigt jedoch zur Einzelstrahlberechnung ein Speicherinterface mit ausreichend hoher Bandbreite was bedeutet, dass eine hohe Datenrate pro Zeiteinheit realisiert werden muss. Das Bearbeiten eines Pixels geschieht durch ein Softwareprogramm, welches auf der Ray-Tracing CPU / RTPU läuft.

Fig. 5 zeigt einen beispielhaften Aufbau einer RTPU. Zu sehen ist ein standard Prozessorkern (RISC Kern) welchem zwei Spezialcoprozessoren parallel geschaltet sind. Die Coprozessoren besitzen jeweils ihre eigenen Register. Es ist jedoch auch möglich durch Spezialbefehle diese Registerinhalte von einem Coprozessor in einen anderen zu überführen. Der Traversierungs-Kern ist ein spezieller Coprozessor, welcher Strahlen effizient durch eine Beschleunigungsstruktur traversieren kann. Hierzu benötigt er ein spezielles Speicherinterface zu den Knoten der Beschleunigungsstruktur (Knoten- Cache). Der Vektor Arithmetik Kern ist ein Spezialcoprozessor, welcher effizient Operationen im 3D Raum durchführen kann. Die von jeder Ray-Tracing Software benötigten Vektoradditionen, Skalarmultiplikationen, Kreuzprodukte und Vektorprodukte können mittels dieser Einheit schnell berechnet werden. Die Vektorarithmetikeinheit benötigt Zugriff auf einen Spezialcache, welcher es ermöglicht ganze Vektoren in einem Takt zu laden.

Die in der RTPU hardwaremäßig implementierten Einzelbefehle oder Logikfunktionen beinhalten die gleichen Algorithmen, welche schon bei der festverdrahteten ersten Ausbildungsform der Erfindung beschrieben sind. Ergänzend hierzu, kann dieser Befehlssatz jedoch um weitere festverdrahtete Befehle und Funktionen ergänzt werden, welche wiederum programmgesteuert zu aktivieren sind. Durch spezielle Traversierungs-Operationen, durch Vektorarithmetik, sowie durch die parallele Abarbeitung mehrer Threads auf einer RTPU wird die für Echtzeitanwendung erforderliche Rechenleistung bereitgestellt und gleichzeitig werden die Auswirkungen der Speicherlatenzen (Wartezeiten auf Speicheranfragen) auf die Systemgeschwindigkeit minimiert oder sogar unrelevant.

Dadurch, dass die RTPU programmgesteuert dazu verwandt werden kann, rekursiv Sekundärstrahlen durch die Szene zu schießen, wird das Programmiermodell, im Vergleich zur festverdrahteten Hardware, erheblich vereinfacht.

Ist die Farbe des Pixels vollständig berechnet, wird diese in den Frame-Buffer und die Distanz in den Z-Buffer geschrieben. Hierdurch ist eine Anbindung an die Standard Rasterisierungshardware möglich.

Beschreibung der Zeichnungen

Fig. 1 zeigt wie aus mehreren Stufen von Objekten einen Baum erstellt werden kann. Zunächst wird als Objekt der Stufe 1 ein Blatt modelliert. Dieses Blatt wird nun mehrfach instantiiert und an einen Ast gesetzt, wodurch ein weiteres Objekt entsteht, jedoch jetzt ein Objekt der Stufe 2. Diese kleinen Äste können nun wieder mehrfach instantiiert werden zu einem Baum als Objekt der Stufe 3.

Fig. 2 zeigt die Ray-Casting-Pipeline (RCP) mit den wichtigsten Datenpfaden, sowie der Schnittstelle zu den Caches.

Fig. 3 zeigt das Top-Level Diagramm einer beispielhaften Implementierung der Erfindung. Die 4 Ray-Casting-Pipelines (RCP) sind über eine Cache-Hierarchie mit den getrennten Speichern für Knoten, Listen und Matrizen verbunden. In diesem Beispieldesign sind je zwei RCP's mit einer Shadingeinheit verbunden, welche Zugriff auf einen lokalen Frame-Buffer und Z-Buffer hat. Über diese werden Farbwerte mittels einer Tone-Mapping-Einheit in einen globalen Frame-Buffer geschrieben und Tiefenwerte direkt in einen globalen Z-Buffer geschickt. An diesen Z-Buffer und Frame-Buffer kann eine dem Stand der Technik entsprechende Rasterisierungshardware (Rasterisierungs Pipeline) angeschlossen werden.

Fig. 4 zeigt die Cache-Infrastruktur, welche die nötige interne Speicherbandbreite im Chip bereitstellt. In der Abbildung handelt es sich um einen binären n-stufigen Cache, es sind jedoch auch andere hierarchische Strukturen denkbar.

Fig. 5 zeigt die beispielhafte Ausführungsform einer Ray-Tracing CPU.

Fig. 6. Beispiel der vereinfachten Geometrie in den Octreeknoten.

Fig. 7. Beispiel der regelmäßigen Grid Beschleunigungsstruktur an einer einfachen Szene. Der Raum ist der Einfachkeit halber nur in 2D gezeichnet.

Fig. 8. Beispiel der k-D Baum Beschleunigungsstruktur an einer einfachen Szene. Der Raum ist der Einfachkeit halber nur in 2D gezeichnet.

Fig. 9. Beispiel der Octree Beschleunigungsstruktur an einer einfachen Szene. Der Raum ist der Einfachkeit halber nur in 2D gezeichnet.

Fig. 10. Zeigt die Dreieckstransformation in den Norm-Dreiecks Raum.

Fig. 11. Zeigt eine beispielhafte Ausbildungsform der Erfindung basierend auf speziellen Ray-Tracing CPU's (RTPU's).

Fig. 12. Vereinfachte Version der in Figure 2 gezeigten Ray-Casting-Pipeline.

Patentansprüche

1. Vorrichtung zur Darstellung von dynamischen, komplexen Szenen, dadurch gekennzeichnet, dass diese aus einem oder mehreren programmgesteuerten Ray-Tracing-Prozessor/en besteht (Fig.5 / Fig.11), dessen/deren Funktionsumfang mittels hardwareimplementierter Traversierungs-Befehle und/oder Vektorarithmetikbefehle und/oder Befehle zur Erstellung von Ray-Tracing Beschleunigungsstrukturen realisiert wird

und/oder dass die Vorrichtung aus hardwarebasierten festverdrahteten oder verknüpften Logikfunktionen besteht (Fig.2 / Fig.3 / Fig.4), welche so organisiert sind, dass auf den Ray-Tracing-Verfahren basierende Funktionen bereitgestellt werden und zu deren Realisierung eine oder mehrere Traversierungs-Einheit/en und eine oder mehrere Listen-Einheit/en und eine oder mehrere Schnittpunktberechnungs-Einheit/en

und ein oder mehrere Transformations-Einheit/en und/oder eine oder mehrere Logikeinheit/en zum Lösen von linearen Gleichungssystemen gebildet werden

und die hardwaremäßige Gestaltung der Ray-Tracing-Prozessoren und/oder der Logikfunktionen in einem oder mehreren programmierbaren Logikbaustein/en oder in festverdrahteter integrierter oder diskreter Logik erfolgt.

2. Vorrichtung gemäß Anspruch 1 dadurch gekennzeichnet, dass die Transformations-Einheit/en funktionell zur Primärstrahlgenerierung und/oder Objektraum-transformation und/oder Normraumtransformation und/oder Reflektions-strahlberechnung und/oder Lichtstrahlberechnung und/oder Normalentransformation verwendet wird.

3. Vorrichtung gemäß Anspruch 1 dadurch gekennzeichnet, dass die Logikeinheit/en zum Lösen linearer Gleichungssysteme für die Objektraumtransformation und/oder Normraumtransformation und/oder Normalentransformation verwendet wird.

4. Vorrichtung gemäß Anspruch 1 dadurch gekennzeichnet, dass die Funktion der Traversierungs-Einheit/en und die hardwaremäßige Umsetzung der Traversierungs-Befehle darauf beruht, dass ein Strahl durch eine Beschleunigungsstruktur, welche auf dem k-D Baum-Verfahren oder dem Octreeverfahren oder dem regelmäßigen Grid-Verfahren basiert, traversiert wird und abhängig von der geometrischen Entfernung des darzustellenden Objektes zur Kamera oder Betrachter, in jedem k-D Baum-Knoten oder Octreeknoten oder Gitterknoten vereinfachte Geometriedaten verwendet werden, sobald der betrachtete Strahl einen Großteil des Volumens des aktuellen k-D Baumknoten oder Octreeknoten oder Gitterknoten überdeckt.

5. Vorrichtung gemäß den Ansprüchen 1 bis 4 dadurch gekennzeichnet, dass die einzelnen Objekte optional mit unterschiedlichem Detaillevel vorliegen und je nach Distanz zur Kamera oder Betrachter mit unterschiedlichen Detaillevels verwendet werden.

6. Vorrichtung gemäß den Ansprüchen 1 bis 4 dadurch gekennzeichnet, dass die Schachtelungstiefe der dargestellten Objekte, beliebig viele Stufen von Objekten in Objekten umfassen kann.

7. Vorrichtung gemäß den Ansprüchen 1 bis 4 dadurch gekennzeichnet, dass jedes Objekt aus jeder Hierarchieebene der Szene von Frame zu Frame verändert, oder dessen geometrische Position und Orientierung verändert werden kann.

8. Vorrichtung gemäß den Ansprüchen 1 bis 7 dadurch gekennzeichnet, dass ein oder mehrere Ray-Tracing Funktionseinheiten in einem oder mehreren integrierten Chip/s hardwaremäßig realisiert werden und diese parallel arbeiten.

9. Vorrichtung gemäß Anspruch 8 dadurch gekennzeichnet, dass die einzelnen Funktionseinheiten über eine n-level Cache-Struktur mit dem Hauptspeicher verbunden sind.

10. Vorrichtung gemäß den Ansprüchen 1 bis 9 dadurch gekennzeichnet, dass die beschriebene Hardware über einen globalen Z-Buffer und Frame-Buffer an eine Standard Rasterisierungshardware angeschlossen ist.

11. Vorrichtung gemäß den Ansprüchen 1 bis 10 dadurch gekennzeichnet dass sich die Logikfunktionen der Rasterisierungshardware auf dem gleichen Chip wie die Logikfunktionen der Ray-Tracing Hardware oder auf einem separaten Chip befinden.

12. Vorrichtung gemäß den Ansprüchen 1 bis 11 dadurch gekennzeichnet, dass die Funktionen der Bilddatengenerierung um die Möglichkeit der Darstellung von dreidimensional wahrnehmbaren dynamischen, komplexen Szenen auf einem zweidimensionalen Display erweitert werden, wobei die Darstellung in zwei horizontal versetzten Teilbildern erfolgt und die Darstellungsform der beiden Teilbilder dem Stand der Technik entspricht.

13. Vorrichtung gemäß den Ansprüchen 1 bis 11 dadurch gekennzeichnet, dass die Funktionen der Bilddatengenerierung um die Möglichkeit der Darstellung von dreidimensional wahrnehmbaren dynamischen, komplexen Szenen auf zwei funktionell zusammenarbeitenden zweidimensionalen Displays erweitert werden, wobei auf jedem Display jeweils ein horizontal versetztes Teilbild dargestellt wird und die Darstellungsform der beiden Teilbilder dem Stand der Technik entspricht.

14. Vorrichtung gemäß den Ansprüchen 1 bis 13 dadurch gekennzeichnet, dass die Bilddatengenerierung und die Darstellung der dynamischen, komplexen Szenen vorzugsweise in Echtzeit erfolgt.

Vorrichtung zur Darstellung von dynamischen komplexen Szenen

Zusammenfassung

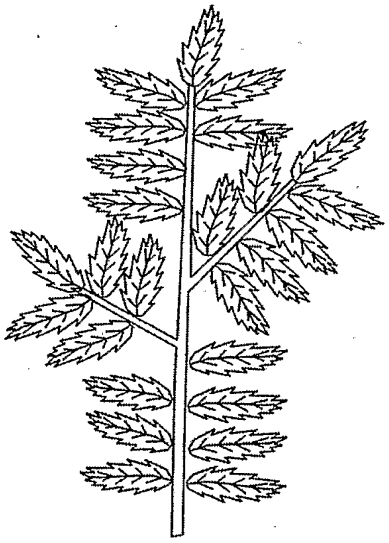
Die Erfindung betrifft eine Vorrichtung, mit welcher dynamische, komplexe dreidimensionale Szenen mit hohen Bildwiederholraten unter Verwendung einer Echtzeit Ray-Tracing Hardwarearchitektur auf einem zweidimensionalen Display dargestellt werden können. Die Erfindung zeichnet sich vor allem dadurch aus, dass sie eine Hierarchiestruktur von beweglichen und veränderbaren Objekten unterstützt, das heißt die Hauptszene kann aus mehreren Objekten bestehen, welche jeweils aus weiteren Objekten aufgebaut sind, wobei diese Schachtelung beliebig fortgeführt werden kann.

Zur erfindungsgemäßen Realisierung wird die Funktion und das Verfahren der bekannten Ray-Tracing-Pipeline um eine Transformations-Einheit, welche die Strahlen in die Objekte hinein transformiert erweitert und diese Funktionen und Verfahren in eine logikbasierte Hardware umgesetzt.

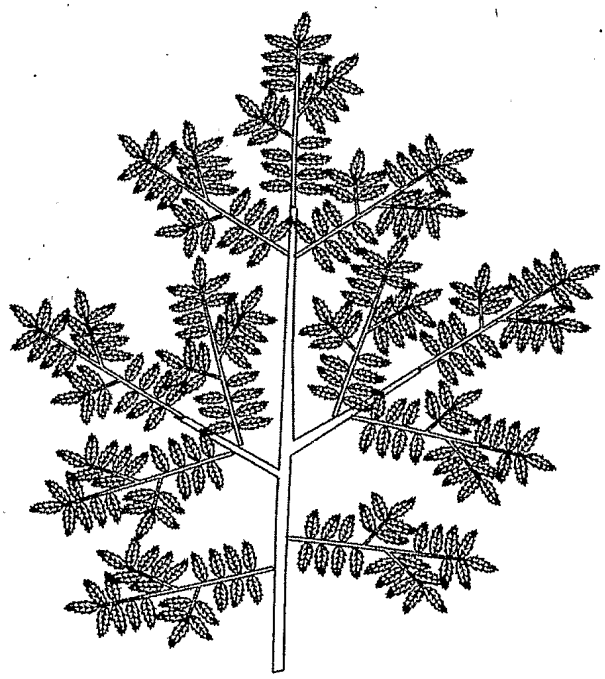
Eine zweite Ausgestaltungsform der Erfindung ermöglicht die Programmierbarkeit des Systems, indem erfindungsgemäß eine neuartige Prozessorarchitektur, bestehend aus der Kombination eines Standard Prozessorkerns mit einem, oder mehreren speziellen Ray-Tracing-Coprozessoren verwendet wird.

Fig. 12 einfügen

Stufe 1 - Blatt



Stufe 2 – Ast



Stufe 3 - Baum

Fig. 1

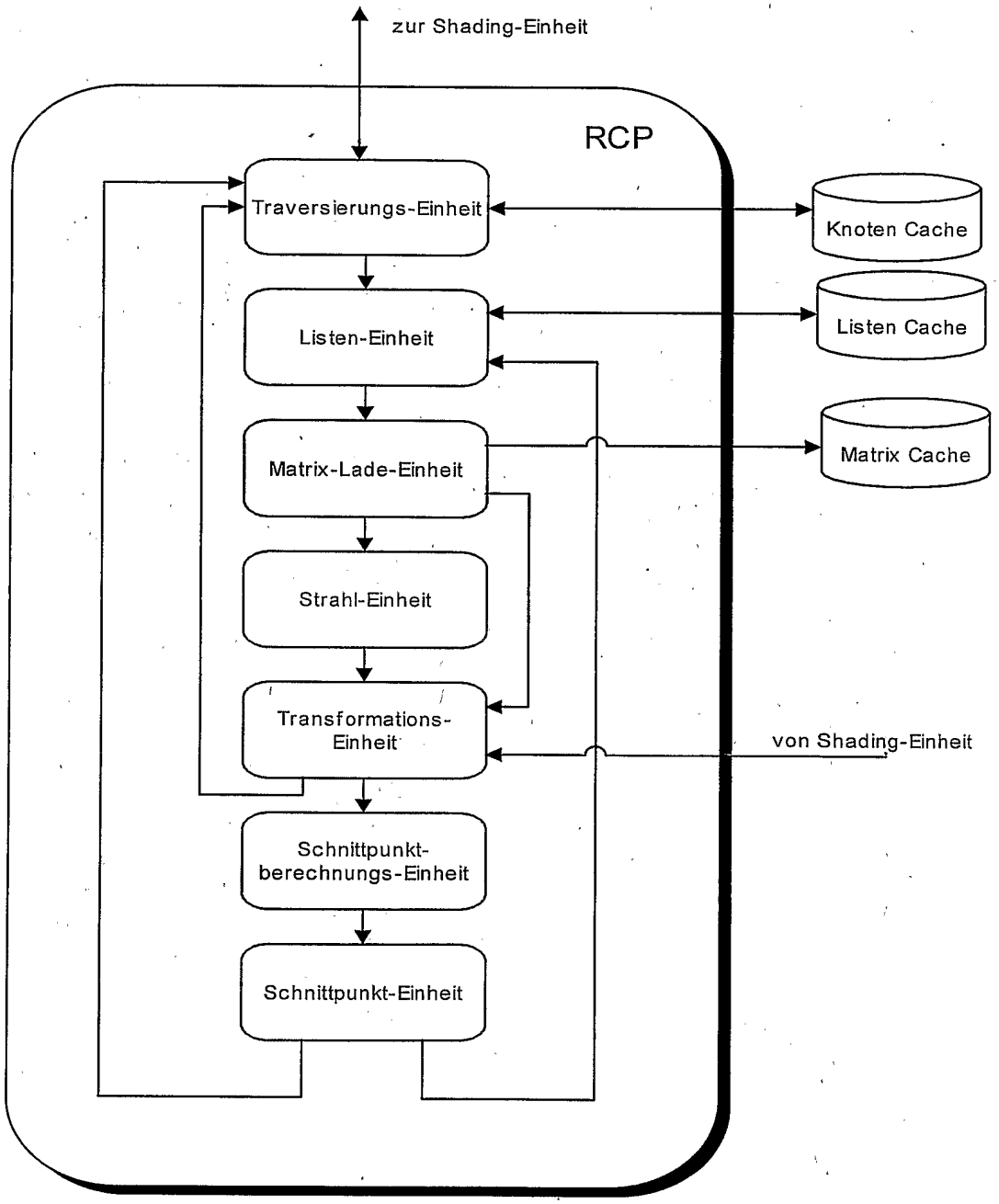

zur Shading-Einheit
RCP
Traversierungs-Einheit
Listen-Einheit
Matrix-Lade-Einheit
Strahl-Einheit
Transformations-Einheit
Schnittpunkt-berechnungs-Einheit
Schnittpunkt-Einheit
Knoten Cache
Listen Cache
Matrix Cache
von Shading-Einheit

Fig. 2

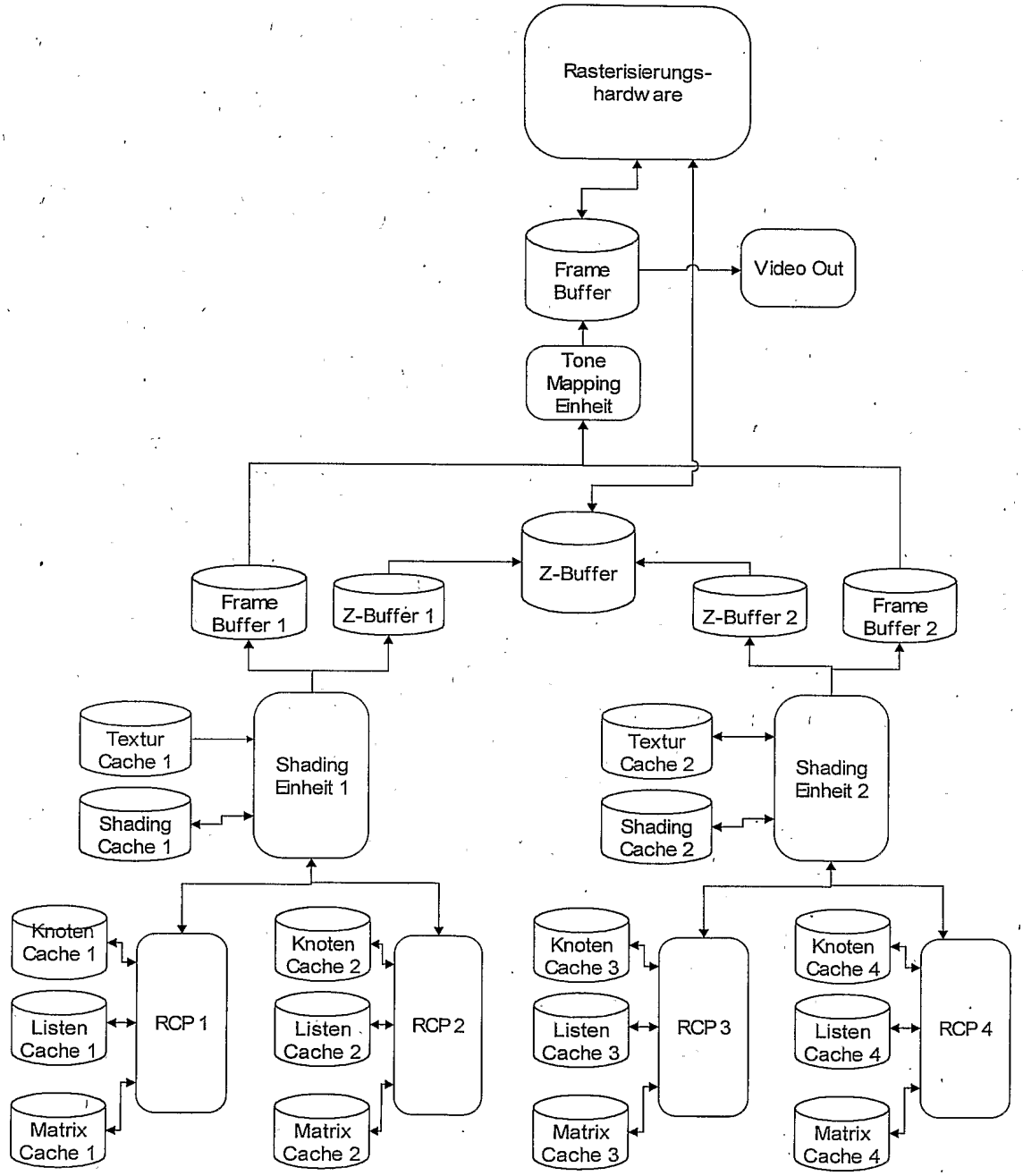

Rasterisierungs-
hardware
Frame
Buffer
Video Out
Tone
Mapping
Einheit
Z-Buffer
Frame
Buffer 1
Z-Buffer 1
Z-Buffer 2
Frame
Buffer 2
Textur
Cache 1
Shading
Cache 1
Shading
Einheit 1
Textur
Cache 2
Shading
Cache 2
Shading
Einheit 2
Knoten
Cache 1
Listen
Cache 1
Matrix
Cache 1
RCP 1
Knoten
Cache 2
Listen
Cache 2
Matrix
Cache 2
RCP 2
Knoten
Cache 3
Listen
Cache 3
Matrix
Cache 3
RCP 3
Knoten
Cache 4
Listen
Cache 4
Matrix
Cache 4
RCP 4

Fig. 3

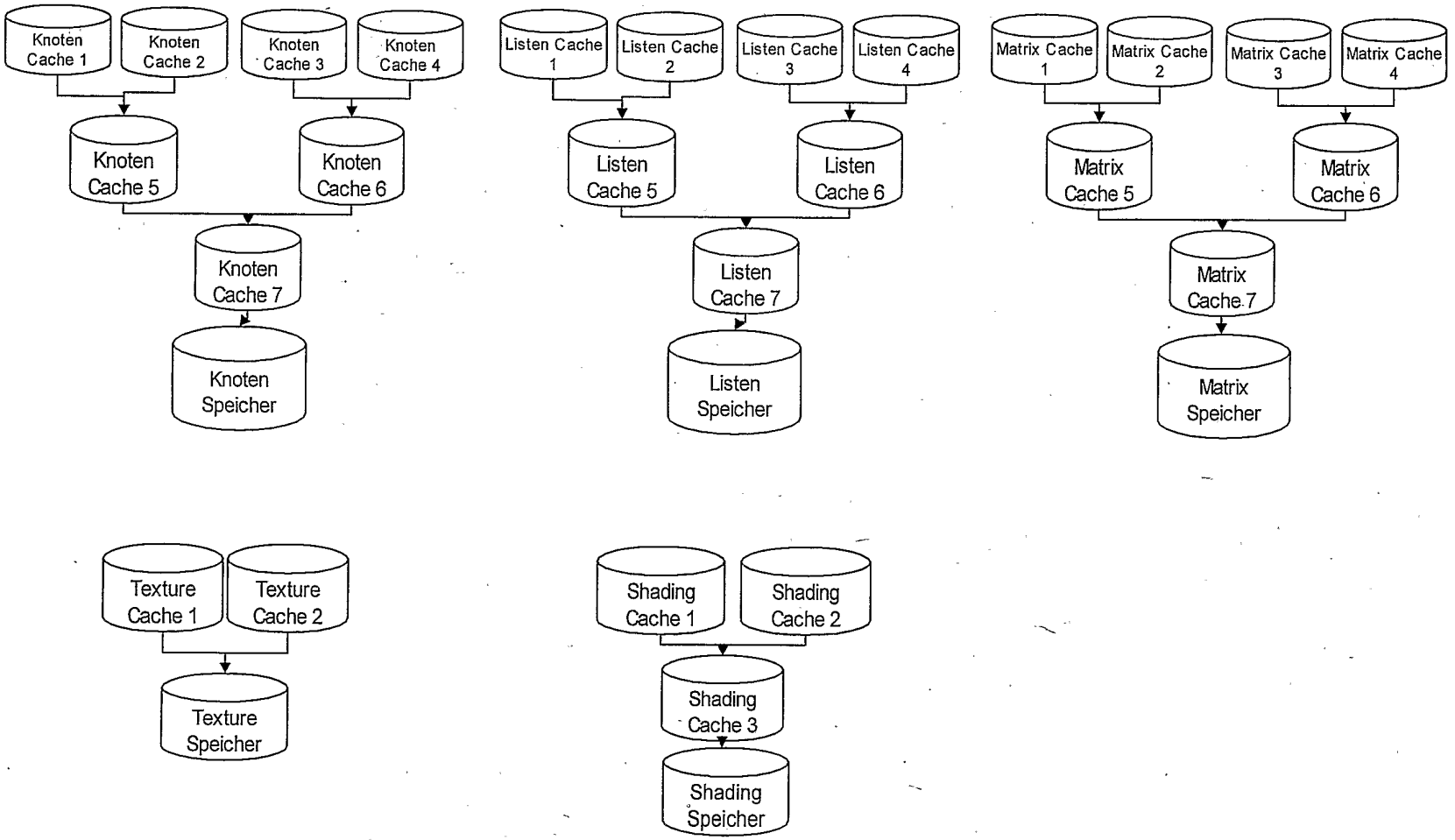

Knoten Cache 1
Knoten Cache 2
Knoten Cache 3
Knoten Cache 4
Knoten Cache 5
Knoten Cache 6
Knoten Cache 7
Knoten Speicher
Listen Cache 1
Listen Cache 2
Listen Cache 3
Listen Cache 4
Listen Cache 5
Listen Cache 6
Listen Cache 7
Listen Speicher
Matrix Cache 1
Matrix Cache 2
Matrix Cache 3
Matrix Cache 4
Matrix Cache 5
Matrix Cache 6
Matrix Cache 7
Matrix Speicher
Texture Cache 1
Texture Cache 2
Texture Speicher
Shading Cache 1
Shading Cache 2
Shading Cache 3
Shading Speicher

Fig. 4

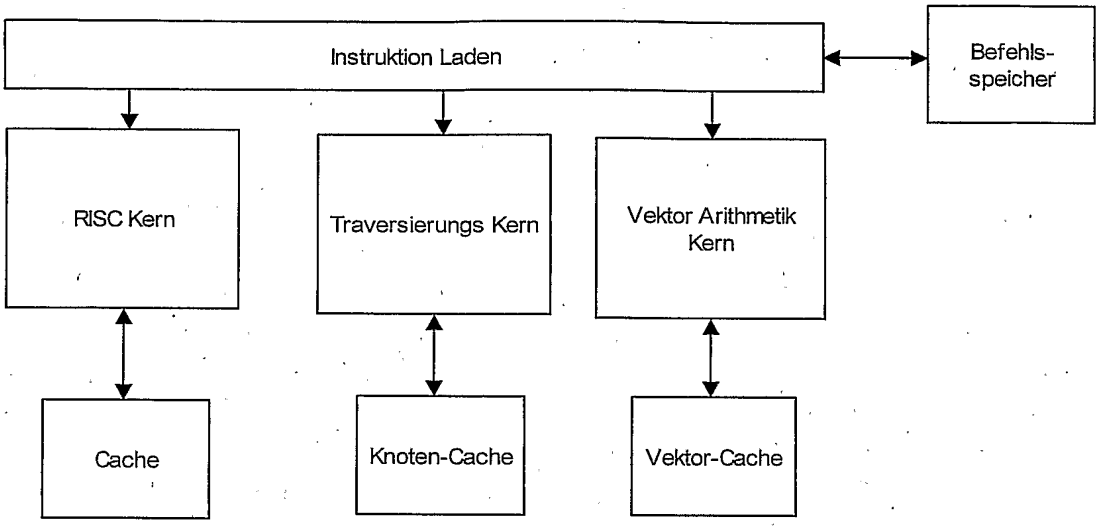

Instruktion Laden
Befehls-speicher
RISC Kern
Traversierungs Kern
Vektor Arithmetik Kern
Cache
Knoten-Cache
Vektor-Cache

Fig. 5

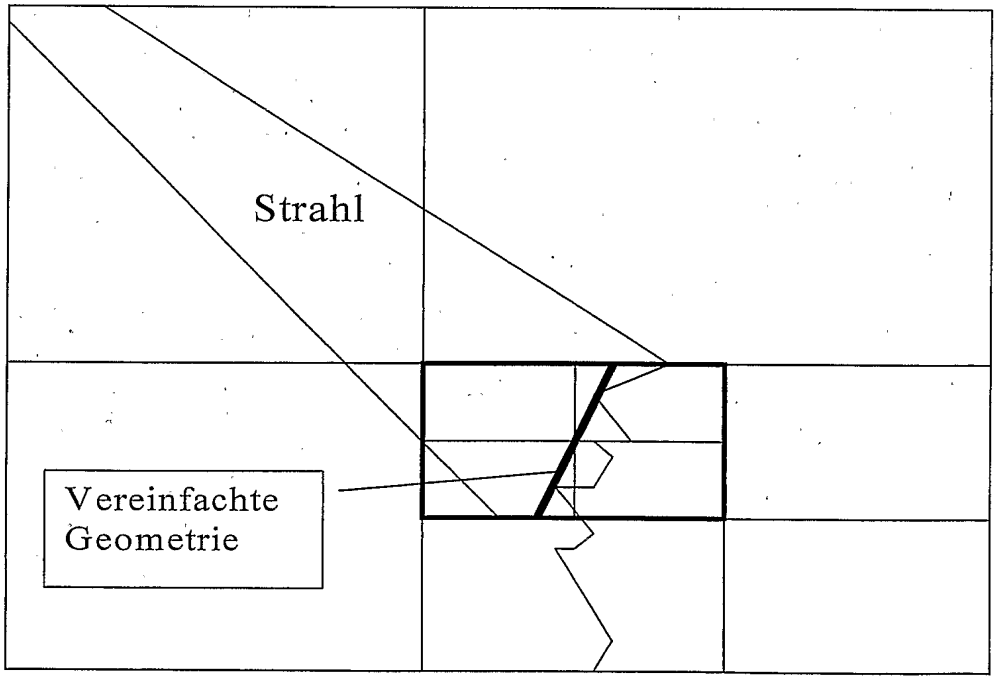

Strahl
Vereinfachte Geometrie

Fig. 6

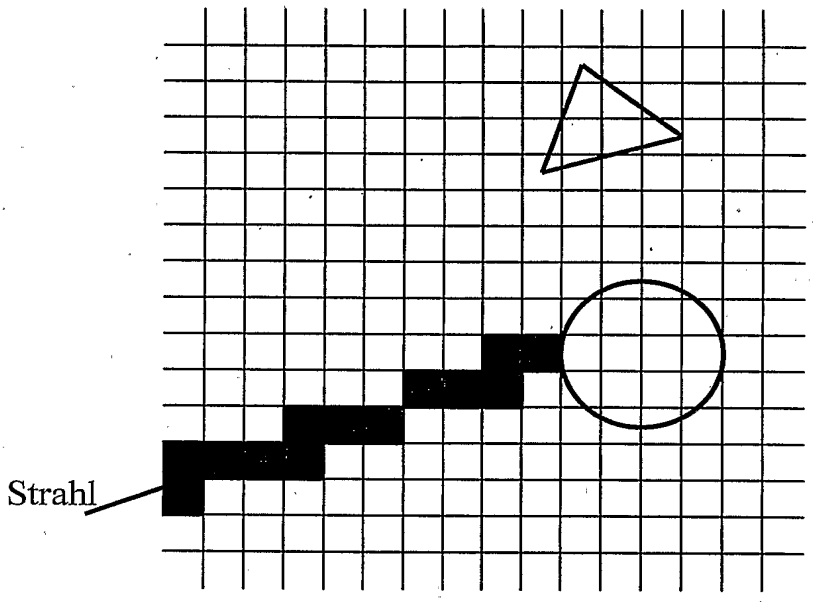


Fig. 7

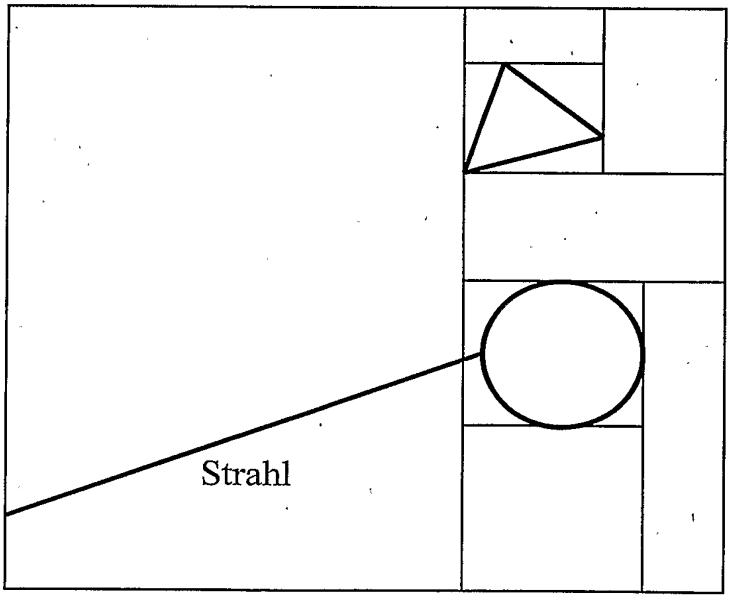


Fig. 8

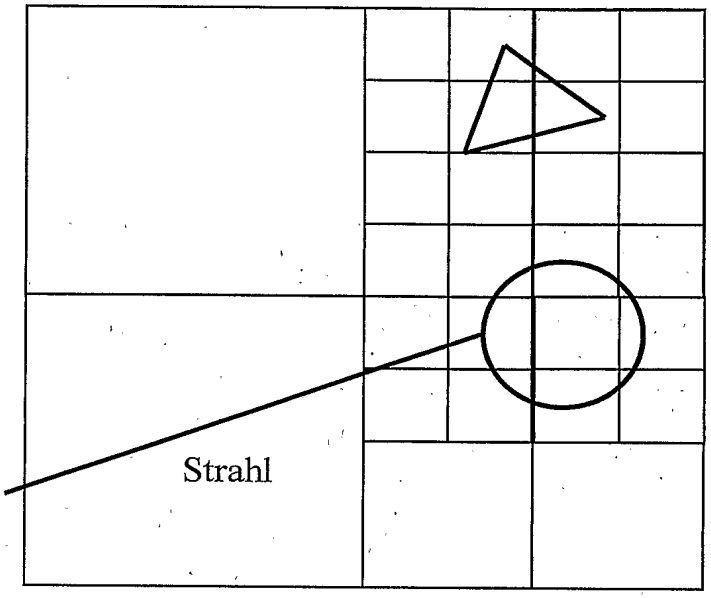

Strahl

Fig. 9

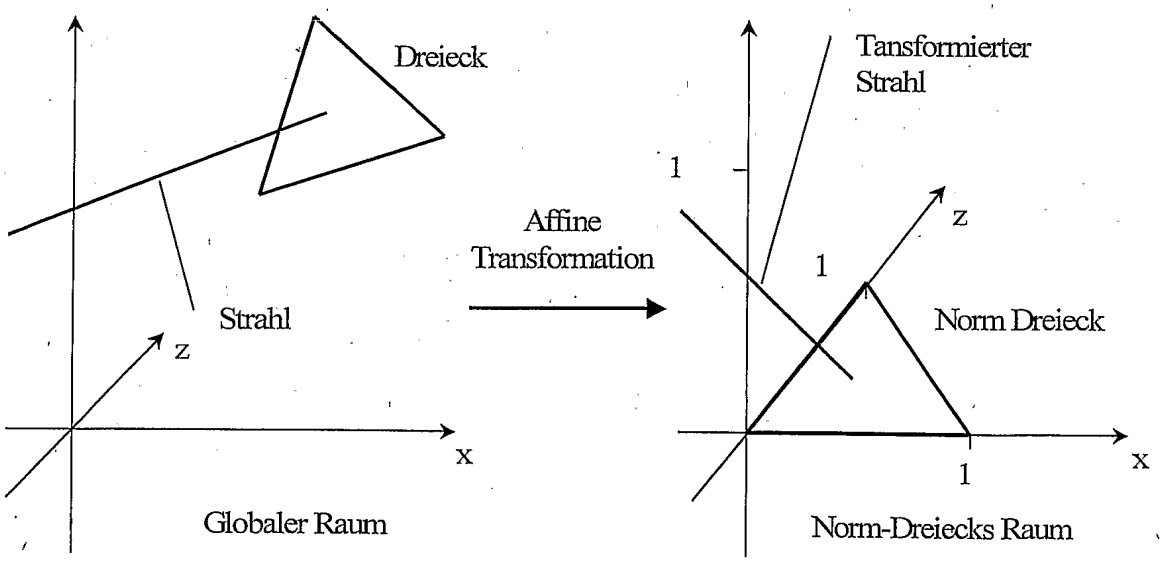

Dreieck
Strahl
z
x
Globaler Raum
Affine
Transformation
Tansformierter
Strahl
1
z
1
Norm Dreieck
1
x
Norm-Dreiecks Raum

Fig. 10

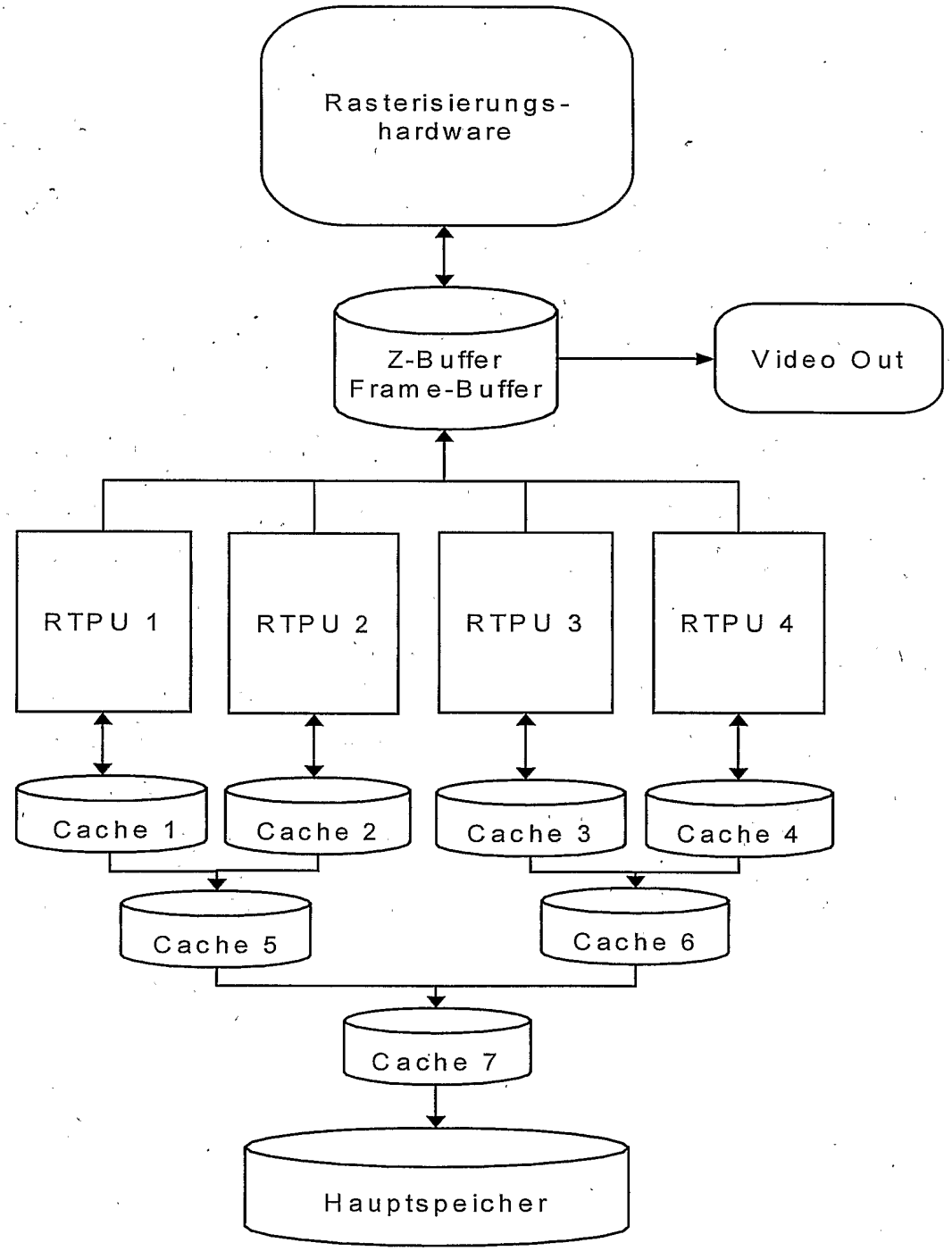
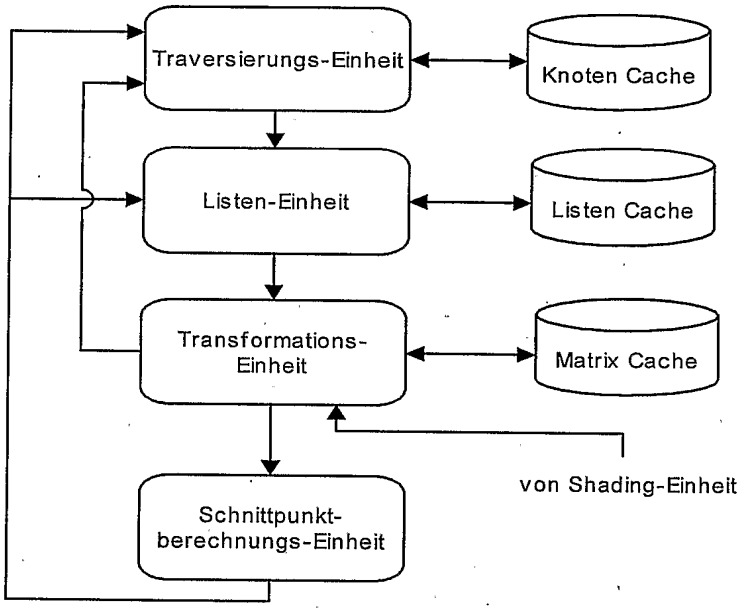

Rasterisierungs-
hardware
Z-Buffer
Frame-Buffer
Video Out
RTPU 1
RTPU 2
RTPU 3
RTPU 4
Cache 1
Cache 2
Cache 3
Cache 4
Cache 5
Cache 6
Cache 7
Hauptspeicher

Fig. 11



Ray-Casting-Pipeline

Fig. 12